C++课程设计报告

# 学生信息管理系统

基于 C++ 和 SQLite 的学生信息管理系统

学院：计算机科学与技术学院

专业：软件工程

班级：软件2101班

学号：20210xxxx

姓名：张三、李四、王五

指导教师：赵老师

完成日期：2025年6月24日

# 2. 任务书

## 课程设计任务

设计并实现一个基于C++的学生信息管理系统，要求具备以下功能：

1. 1. 学生信息管理：实现学生信息的添加、删除、修改、查询等功能
2. 2. 课程信息管理：实现课程信息的添加、删除、修改、查询等功能
3. 3. 成绩信息管理：实现成绩信息的录入、修改、查询、统计等功能
4. 4. 用户管理：实现用户登录、注册、权限管理等功能
5. 5. 数据存储：使用SQLite数据库实现数据的持久化存储

## 技术要求

1. 1. 采用面向对象的设计方法，使用C++语言实现
2. 2. 使用SQLite数据库进行数据存储
3. 3. 实现图形用户界面，可使用Qt框架
4. 4. 代码结构清晰，注释完整，具有良好的可维护性
5. 5. 实现基本的数据安全和用户权限控制

## 完成时间

课程设计周期为4周，要求在2025年6月30日前完成并提交报告。

## 考核方式

1. 1. 系统功能完整性：40%
2. 2. 技术实现水平：30%
3. 3. 代码质量与规范：20%
4. 4. 文档完整性：10%

指导教师：赵老师

日期：2025年5月27日

# 3. 目录

1. 封面

2. 任务书

3. 目录

4. 前言，需求分析

4.1 项目背景

4.2 需求分析

4.3 可行性分析

5. 系统设计

5.1 总体架构设计

5.2 数据库设计

5.3 功能模块设计

5.4 界面设计

5.5 安全设计

6. 系统实现

6.1 开发环境

6.2 学生信息管理模块实现

6.3 课程信息管理模块实现

6.4 成绩信息管理模块实现

6.5 用户管理模块实现

6.6 数据库操作实现

6.7 界面实现

7. 系统测试

7.1 测试环境

7.2 功能测试

7.3 性能测试

7.4 兼容性测试

7.5 测试结果分析

8. 总结与展望

8.1 项目总结

8.2 技术亮点

8.3 存在问题

8.4 未来展望

9. 主要源代码

9.1 核心类设计

9.2 数据库操作类

9.3 业务逻辑类

9.4 界面实现类

9.5 工具类

# 4. 前言，需求分析

## 4.1 项目背景

（由张三完成）

随着信息技术的发展和教育信息化的推进，学生信息管理系统已成为教育机构管理学生信息的重要工具。传统的纸质档案管理方式已经无法满足现代教育管理的需求，开发一个高效、便捷的学生信息管理系统具有重要意义。

本项目旨在开发一个基于C++和SQLite的学生信息管理系统，实现学生基本信息、课程信息和成绩信息的综合管理，为教育机构提供一个现代化的信息管理平台，同时也为C++课程设计提供一个综合性的实践项目。

## 4.2 需求分析

（由张三完成）

根据用户需求调研，本系统需要实现以下核心功能：

### 4.2.1 功能需求

1. 1. 学生信息管理

* - 学生信息添加：添加新学生的基本信息
* - 学生信息修改：修改已有学生的信息
* - 学生信息删除：删除学生信息
* - 学生信息查询：按学号、姓名、班级等条件查询学生信息

1. 2. 课程信息管理

* - 课程信息添加：添加新课程信息
* - 课程信息修改：修改已有课程信息
* - 课程信息删除：删除课程信息
* - 课程信息查询：按课程编号、名称等条件查询课程信息

1. 3. 成绩信息管理

* - 成绩录入：录入学生课程成绩
* - 成绩修改：修改已有成绩
* - 成绩查询：按学号、课程等条件查询成绩
* - 成绩统计：统计平均分、最高分、最低分等
* - 成绩排名：按成绩进行排名

1. 4. 用户管理

* - 用户登录：用户登录系统
* - 用户注册：新用户注册
* - 密码修改：修改用户密码
* - 权限管理：不同用户具有不同的操作权限

### 4.2.2 非功能需求

1. 1. 性能需求

* - 响应时间：用户操作响应时间不超过1秒
* - 并发处理：支持多用户同时操作
* - 数据容量：支持存储至少10000条学生记录

1. 2. 安全需求

* - 用户认证：用户登录需要用户名和密码验证
* - 权限控制：不同角色用户具有不同的操作权限
* - 数据备份：支持数据备份和恢复功能
* - 操作日志：记录重要操作的日志

1. 3. 可用性需求

* - 界面友好：操作界面简洁直观
* - 易学易用：操作简单，有必要的提示信息
* - 容错性：对用户误操作有适当的提示和处理

## 4.3 可行性分析

（由李四完成）

### 4.3.1 技术可行性

本项目采用C++语言开发，结合SQLite数据库进行数据存储，使用Qt框架实现图形用户界面。这些技术都是成熟的、稳定的技术，团队成员在课程学习中已经掌握了这些技术的基本应用，具备开发此系统的技术能力。

C++语言具有高效、面向对象的特点，适合开发此类管理系统；SQLite是一个轻量级的数据库，适合小型应用的数据存储；Qt框架提供了丰富的UI组件和跨平台能力，可以快速构建图形界面。

### 4.3.2 经济可行性

本项目作为课程设计，主要目的是学习和实践，不涉及商业运营，因此不需要考虑投资回报等经济因素。开发过程中使用的技术和工具均为开源免费软件，不需要额外的软件许可费用。

### 4.3.3 操作可行性

系统设计注重用户体验，界面简洁直观，操作流程符合用户习惯，降低了用户的学习成本。同时，系统提供详细的操作提示和帮助信息，确保用户能够顺利完成各项操作。

# 5. 系统设计

## 5.1 总体架构设计

（由李四完成）

本系统采用经典的三层架构设计，包括表示层、业务逻辑层和数据访问层，实现了系统的高内聚、低耦合。

### 5.1.1 系统架构图

+------------------+  
| 表示层(UI) |  
| Qt界面 |  
+--------+---------+  
 |  
+--------+---------+  
| 业务逻辑层(BLL) |  
| 业务处理类 |  
+--------+---------+  
 |  
+--------+---------+  
| 数据访问层(DAL) |  
| 数据库操作 |  
+--------+---------+  
 |  
+--------+---------+  
| 数据库 |  
| SQLite |  
+------------------+

### 5.1.2 技术架构

* - 开发语言：C++
* - 界面框架：Qt
* - 数据库：SQLite
* - 开发工具：Qt Creator
* - 版本控制：Git

## 5.2 数据库设计

（由李四完成）

### 5.2.1 ER图

系统的核心实体包括学生(Student)、课程(Course)、成绩(Grade)和用户(User)，它们之间的关系如下：

* - 一个学生可以选修多门课程
* - 一门课程可以被多个学生选修
* - 一个学生在一门课程上有一个成绩
* - 用户可以是管理员或普通用户

### 5.2.2 数据库表设计

1. 1. 学生表(students)

CREATE TABLE students (  
 student\_id TEXT PRIMARY KEY,  
 name TEXT NOT NULL,  
 gender TEXT CHECK(gender IN ('男', '女')),  
 age INTEGER,  
 class TEXT,  
 department TEXT,  
 phone TEXT,  
 email TEXT,  
 address TEXT,  
 create\_time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
);

1. 2. 课程表(courses)

CREATE TABLE courses (  
 course\_id TEXT PRIMARY KEY,  
 name TEXT NOT NULL,  
 credit REAL NOT NULL,  
 hours INTEGER NOT NULL,  
 teacher TEXT,  
 description TEXT,  
 create\_time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
);

1. 3. 成绩表(grades)

CREATE TABLE grades (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 student\_id TEXT NOT NULL,  
 course\_id TEXT NOT NULL,  
 score REAL NOT NULL,  
 semester TEXT NOT NULL,  
 create\_time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 FOREIGN KEY (student\_id) REFERENCES students(student\_id),  
 FOREIGN KEY (course\_id) REFERENCES courses(course\_id),  
 UNIQUE(student\_id, course\_id, semester)  
);

1. 4. 用户表(users)

CREATE TABLE users (  
 username TEXT PRIMARY KEY,  
 password TEXT NOT NULL,  
 role TEXT CHECK(role IN ('admin', 'user')) DEFAULT 'user',  
 create\_time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
);

## 5.3 功能模块设计

（由王五完成）

系统功能模块主要分为以下几个部分：

### 5.3.1 学生信息管理模块

学生信息管理模块负责学生基本信息的管理，包括：

* - 添加学生：输入学生信息，添加新学生
* - 修改学生：修改已有学生的信息
* - 删除学生：删除学生信息
* - 查询学生：按条件查询学生信息
* - 导入导出：支持批量导入导出学生信息

### 5.3.2 课程信息管理模块

课程信息管理模块负责课程信息的管理，包括：

* - 添加课程：输入课程信息，添加新课程
* - 修改课程：修改已有课程的信息
* - 删除课程：删除课程信息
* - 查询课程：按条件查询课程信息

### 5.3.3 成绩信息管理模块

成绩信息管理模块负责学生成绩的管理，包括：

* - 录入成绩：录入学生的课程成绩
* - 修改成绩：修改已有成绩
* - 查询成绩：按学生、课程等条件查询成绩
* - 统计成绩：统计平均分、最高分、最低分等
* - 成绩排名：按成绩进行排名
* - 成绩分析：生成成绩分析报表

### 5.3.4 用户管理模块

用户管理模块负责系统用户的管理，包括：

* - 用户登录：验证用户身份
* - 用户注册：新用户注册
* - 密码修改：修改用户密码
* - 权限管理：设置用户权限

## 5.4 界面设计

（由王五完成）

系统界面采用Qt框架实现，主要包括以下几个界面：

### 5.4.1 登录界面

登录界面是系统的入口，用户需要输入用户名和密码进行身份验证。界面包含以下元素：

* - 用户名输入框
* - 密码输入框
* - 登录按钮
* - 注册按钮

### 5.4.2 主界面

主界面是系统的核心界面，包含系统的主要功能模块。界面采用菜单栏和工具栏的组合方式，包含以下元素：

* - 菜单栏：包含文件、编辑、视图、工具、帮助等菜单
* - 工具栏：包含常用操作的快捷按钮
* - 状态栏：显示系统状态信息
* - 导航栏：用于切换不同的功能模块
* - 内容区：显示当前模块的内容

### 5.4.3 学生管理界面

学生管理界面用于管理学生信息，包含以下元素：

* - 学生列表：显示所有学生信息
* - 搜索框：用于搜索学生
* - 添加按钮：添加新学生
* - 修改按钮：修改选中学生的信息
* - 删除按钮：删除选中的学生
* - 导入导出按钮：导入导出学生信息

## 5.5 安全设计

（由张三完成）

为保证系统的安全性，采取了以下安全措施：

### 5.5.1 用户认证

系统采用用户名和密码的方式进行用户认证，密码使用MD5加密存储，确保密码安全。

### 5.5.2 访问控制

系统实现了基于角色的访问控制，不同角色的用户具有不同的操作权限：

* - 管理员：具有所有操作权限
* - 普通用户：只具有查询权限，不能进行添加、修改、删除等操作

### 5.5.3 数据安全

系统采取了以下措施保证数据安全：

* - 数据备份：定期备份数据库
* - 数据恢复：支持从备份恢复数据
* - 输入验证：对用户输入进行验证，防止SQL注入等攻击
* - 日志记录：记录重要操作的日志，便于追踪问题

# 6. 系统实现

## 6.1 开发环境

（由张三完成）

### 6.1.1 硬件环境

* - CPU: Intel Core i7 或同等性能
* - 内存: 16GB 或以上
* - 硬盘: 256GB SSD 或以上

### 6.1.2 软件环境

* - 操作系统: Windows 10/11 或 macOS 或 Linux
* - 开发工具: Qt Creator 6.0+
* - 编译器: MinGW 8.1+ 或 MSVC 2019+ 或 GCC 9.0+
* - 数据库: SQLite 3.35+
* - 版本控制: Git 2.30+

### 6.1.3 开发工具

* - Qt Creator: 用于C++代码编写和UI设计
* - Qt Designer: 用于界面设计
* - SQLite Browser: 用于数据库设计和测试
* - Git: 用于版本控制
* - Visual Studio Code: 用于辅助编码

## 6.2 学生信息管理模块实现

（由李四完成）

### 6.2.1 学生类设计

class Student {  
private:  
 QString studentId;  
 QString name;  
 QString gender;  
 int age;  
 QString className;  
 QString department;  
 QString phone;  
 QString email;  
 QString address;  
 QDateTime createTime;  
  
public:  
 Student();  
 Student(const QString &studentId, const QString &name, const QString &gender,   
 int age, const QString &className, const QString &department,  
 const QString &phone, const QString &email, const QString &address);  
  
 // Getters and setters  
 QString getStudentId() const;  
 void setStudentId(const QString &studentId);  
 QString getName() const;  
 void setName(const QString &name);  
 // ... other getters and setters  
  
 // Database operations  
 bool save();  
 bool update();  
 bool remove();  
 static Student findById(const QString &studentId);  
 static QList<Student> findAll();  
 static QList<Student> findByName(const QString &name);  
 static QList<Student> findByClass(const QString &className);  
};

### 6.2.2 学生管理界面实现

class StudentManagementWidget : public QWidget {  
 Q\_OBJECT  
  
private:  
 QTableView \*tableView;  
 QStandardItemModel \*model;  
 QLineEdit \*searchEdit;  
 QPushButton \*addButton;  
 QPushButton \*editButton;  
 QPushButton \*deleteButton;  
 QPushButton \*importButton;  
 QPushButton \*exportButton;  
  
 void setupUI();  
 void loadData();  
 void setupConnections();  
  
private slots:  
 void onAddButtonClicked();  
 void onEditButtonClicked();  
 void onDeleteButtonClicked();  
 void onImportButtonClicked();  
 void onExportButtonClicked();  
 void onSearchTextChanged(const QString &text);  
  
public:  
 explicit StudentManagementWidget(QWidget \*parent = nullptr);  
 ~StudentManagementWidget();  
};

## 6.3 课程信息管理模块实现

（由王五完成）

### 6.3.1 课程类设计

class Course {  
private:  
 QString courseId;  
 QString name;  
 double credit;  
 int hours;  
 QString teacher;  
 QString description;  
 QDateTime createTime;  
  
public:  
 Course();  
 Course(const QString &courseId, const QString &name, double credit,   
 int hours, const QString &teacher, const QString &description);  
  
 // Getters and setters  
 QString getCourseId() const;  
 void setCourseId(const QString &courseId);  
 QString getName() const;  
 void setName(const QString &name);  
 // ... other getters and setters  
  
 // Database operations  
 bool save();  
 bool update();  
 bool remove();  
 static Course findById(const QString &courseId);  
 static QList<Course> findAll();  
 static QList<Course> findByName(const QString &name);  
 static QList<Course> findByTeacher(const QString &teacher);  
};

### 6.3.2 课程管理界面实现

class CourseManagementWidget : public QWidget {  
 Q\_OBJECT  
  
private:  
 QTableView \*tableView;  
 QStandardItemModel \*model;  
 QLineEdit \*searchEdit;  
 QPushButton \*addButton;  
 QPushButton \*editButton;  
 QPushButton \*deleteButton;  
  
 void setupUI();  
 void loadData();  
 void setupConnections();  
  
private slots:  
 void onAddButtonClicked();  
 void onEditButtonClicked();  
 void onDeleteButtonClicked();  
 void onSearchTextChanged(const QString &text);  
  
public:  
 explicit CourseManagementWidget(QWidget \*parent = nullptr);  
 ~CourseManagementWidget();  
};